```
 1  #
 2  # procorder.pl (c) 1998,1999 E-Dialog, Inc. All Rights Reserved.
 3  #
 4  # PURPOSE DISCUSSION
 5  #
 6  # program accepts a raw email stream from MS Exchange Server via export to ADO-compliant datastore
 7  # then uses email body preprocessing and extraction rules - tailored per campaign in ./cf/ - to construct order reports.
 8  #
 9  # cf files are campaign optimized:
10  #
11  # mailingDBLayout.cf                   -> field mappings for a campaign's master mailing table
12  # messageBodyDataLayout.cf             -> DSNs, field mappings, critical rule designations, order-designation bounds, synonymn table
13  # messageBodyIndicateAddressChange.cf  -> rulesets for indicating ADDRESS CHANGES
14  # messageBodyIndicateBuyAll.cf         -> rulesets for indicating BUY ALL state
15  # messageBodyPreProcRules.cf           -> body text preprocessor configuration and rulesets
16  #
17  # program produces diagnostic and order-entry bound output files and logs outcomes of all ruleset evaluations leading to
18  # preparation of those output files, which include:
19  #
20  # <$ARGV[1]>           -> summary order report file (designed for data entry, faxing, emailing to fulfillment)
21  # EXC_<...>            -> (companion) summary exception report (same format as s. order report with annotations to indicate rule failures for man
ual review)
22  # OUT_<...>            -> detailed order stream file (all fields discretely mapped and normalized for input into batch postprocessor)
23  # EXC_DETAIL_<...>     -> (companion)..exceptions..discrete fields, etc.
24  # EXC_BODY_<...>       -> annotated diagnostic report containing all message bodies - sans original message - for aid in resolving exceptions man
ually
25  #
26  # output files are (default) tab-delimited text
27  #
28  # PROGRAM REQUIRES
29  #
30  #       Perl 5.003 interpreter later compiled for WIN32
31  #       MS ADO components 1.5 (2.0+ recommended)
32  #
33  # RELEASE HISTORY
34  #
35  # 981020AE: 0.5    first usage: to extract caro110 order reports
36  # 981204AE: 0.9    upgrade to handle caro112, multiple item selections
37  # 981208AE: 1.1    improve flagging
38  # 981211AE: 1.5    adapt to extract IDEAS@WORK add/changes
39  # 981211AE: 3.1    fix bugs in EXC_BODY reporting / enhance it
40  # 981222AE: 2.4    add cfPATH support
41  # 981223AE: 2.4.1  cfPath logging; fix synonymn (SUBJECT TOKEN) lookup <trailing \s*>
42  # 981223AE: 2.5    reorder reporting fields
43  # 981224AE: 2.5.1  reorder reporting fields
44  # 981225AE: 2.5.2  update SHIPPING_ADDRESS fields to reflect presence of T,B & S diagnostic fields
45  # 981226AE: 2.6    code review/dox; update cf to require MAILINGID
46  # 990106AE: 2.6.5  standardize on LOG ID for use in address change
47  # 990129ae: 2.7    add support for separate billing, shipping addresses
48  # 990201ae: 3.0    add record-recovery via ADO -> mailing database lookups
49  # 990202ae: 3.0.1  record-update ADO -> mailing database
50  # 990202ae: 3.0.2  fix leading digits bug in getDateTimestamp ftn
51  # 990203ae: 3.1    fix MDB query bugs; introduce detection of SELECT && UPDATE SUCCESS; verify decision-tree based UPDATES
52  # 990205ae: 3.1.1  fix address3 bug; provide regression compatibility for /ADDCHANGE
53  # 990206ae: 3.1.2  throw .noTOK as a HARD_EXC
54
55  ################################################### INCL ###################################################
56
57  use Win32::OLE;
58
59  ################################################### ENV ###################################################
60
61  $grequiredARG    = 4;
62  $grevision       = '3.1.2';
63
64  if(scalar(@ARGV) < $grequiredARG) { die "($grevision) Usage: procOrder.pl [userName] [reportName] [cfPATH <campaignMnemonic>] [ <buyToken> | /multi | /add
change ]\n"; }
65
66  $guser           = $ARGV[0];
```

```perl
 67 $GreportFile            = $ARGV[1];
 68 $GcfPath                = $ARGV[2];
 69 $Gsubj_buyToken         = $ARGV[3];
 70 $GexcReportFile         = 'EXC_'.$GreportFile;
 71 $GexcBODYReportFile     = 'EXC_BODY_'.$GreportFile;
 72 $GoutFile               = 'OUT_'.$GreportFile;
 73 $GexceptionsFile        = 'EXC_DETAIL_'.$GreportFile;
 74
 75 if($ARGV[4] == 1)  { $GDEBUG = 1; }  else {$GDEBUG = 0;}
 76 if($ARGV[4] == 2)  { $LGDEBUG_FTN = 1; } else {$LGDEBUG_FTN = 0; }
 77 #######################################################
 78 ###################### DEFN ###########################
 79
 80 ## MISC GLOBALS ##
 81
 82 $Gplatform              = 'WIN';
 83 $GeventType             = 'INFO';
 84 $GbasePath              = '';
 85 $GfieldDelimiter        = '\t';
 86 $Gspace                 = ' ';
 87 $Gnewline               = '\n';
 88 $Gcomma                 = ',';
 89 $Gpipe                  = '|';
 90 $GeventRequiresFUP      = '';
 91 $GmultiItemOrderSwitch  = '/MULTI';
 92 $GaddChangeSwitch       = '/ADDCHANGE';
 93
 94 ################ OUTPUT FILE LAYOUTS ################
 95
 96 ## DO NOT change alpha ordering of lowercase field names (e.g .. t_)
 97 ## This is vital in the BODY field layout defn file since the rules reference the letter prefixes
 98
 99 ## reportFile (note: EXC_reportFileName also written from this template)
100
101 %GreportFileRecord = (
102     a_LID                   => '_e_$thisLogkey',
103     ab_LINE                 => '<fill-in>',
104     b_RID                   => '_e_$thisRecordID',
105     c_THIS_SUBJECT          => '_e_$thisSubject',
106     d_THIS_BODY_BOTTOM      => '_e_$thisBodyBottom',
107     e_THIS_BODY_TOP         => '_e_$thisBodyTop',
108     ia_FUP_REQUIRED         => '_e_$eventRequiresFUP',
109     ib_LOCATION_BUY_TOKEN   => '_e_$thisBuyStatus',
110     j_REVIEW_ACTION_TAKEN   => '<review>',
111     za_SELECTED_ITEMS       => '_e_$thisItemsSelectedstack',
112     zd_PRIORITY             => '_e_$thisMailingID',
113     zf_CUSTNO               => '_e_($thisCustNo==-1 || $thisCustNo == 0)?"":$thisCustNo',
114     zi_LISTID               => '_e_($thisListID==-1)?"":$thisListID',
115     zm_FROM_ADDRESS         => '_e_$thisFromAddress',
116     zp_SPECIAL_INSTRUCTIONS => '<none>',
117     zr_PHONE                => '_e_$results{\`r_bphone\`}{}',
118     zx_BILLING_ADDRESS      => '_e_$thisBillingAddressBlock',
119     zy_SHIPPING_ADDRESS     => '_e_$thisShippingAddressBlock',
120     zz_MDB_BILLING_ADDRESS  => '_e_$thisMDBbillingAddressBlock',
121
122 );
123
124
125 ## outfile
126
127 %GoutFileRecord = (
128     a_LID               => '_e_$thisLogkey',
129     aa_RID              => '_e_$thisRecordID',
130     ab_THIS_SUBJECT     => '_e_$thisSubject',
131     ac_THIS_BODY_BOTTOM => '_e_$thisBodyBottom',
132     ad_THIS_BODY_TOP    => '_e_$thisBodyTop',
133     b_LISTID            => '_e_$thisListID',
134     hh_CUSTNO           => '_e_$thisCustNo',
135
```

```perl
136         '.
137         C_PRIORITY              => '_e_$thismailingID',
138         e_bfirst                => '_e_$results{'_e_bfirst'}',
139         f_bmiddle               => '_e_$results{'f_bmiddle'}',
140         g_blast                 => '_e_$results{'g_blast'}',
141         h_btitle                => '_e_$results{'h_btitle'}',
142         i_bcompany              => '_e_$results{'i_bcompany'}',
143         j_bdepartment           => '_e_$results{'j_bdepartment'}',
144         k_baddress1             => '_e_$results{'k_baddress1'}',
145         l_baddress2             => '_e_$results{'l_baddress2'}',
146         m_baddress3             => '_e_$results{'m_baddress3'}',
147         n_bcity                 => '_e_$results{'n_bcity'}',
148         o_bstate                => '_e_$results{'o_bstate'}',
149         p_bpostal               => '_e_$results{'p_bpostal'}',
150         q_bcountry              => '_e_$results{'q_bcountry'}',
151         r_bphone                => '_e_$results{'r_bphone'}',
152         s_bfax                  => '_e_$results{'s_bfax'}',
153         t_bemail                => '_e_$results{'t_bemail'}',
154
155         ue_sfirst               => '_e_$results{'ue_sfirst'}',
156         uf_smiddle              => '_e_$results{'uf_smiddle'}',
157         ug_slast                => '_e_$results{'ug_slast'}',
158         uh_stitle               => '_e_$results{'uh_stitle'}',
159         ui_scompany             => '_e_$results{'ui_scompany'}',
160         uj_sdepartment          => '_e_$results{'uj_sdepartment'}',
161         uk_saddress1            => '_e_$results{'uk_saddress1'}',
162         ul_saddress2            => '_e_$results{'ul_saddress2'}',
163         um_saddress3            => '_e_$results{'um_saddress3'}',
164         un_scity                => '_e_$results{'un_scity'}',
165         uo_sstate               => '_e_$results{'uo_sstate'}',
166         up_spostal              => '_e_$results{'up_spostal'}',
167         uq_scountry             => '_e_$results{'uq_scountry'}',
168         ur_sphone               => '_e_$results{'ur_sphone'}',
169         us_sfax                 => '_e_$results{'us_sfax'}',
170         ut_semail               => '_e_$results{'ut_semail'}',
171         ux_FROM_ADDRESS         => '_e_$thisFromAddress',
172         uy_LOCATION_BUY_TOKEN   => '_e_$thisBuyStatus',
173         v_SUBJECT               => '_e_$thisSubject',
174         z_FUP_REQUIRED          => '_e_$eventRequiresFUP',
175         zz_REVIEW_TYPE          => '_e_$EXCEPTION_TYPE',
176
177         );
178
179         ## exceptionsFile - BODY only
180         %exceptionsBODYfileRecord = (
181
182
183         a_LID                   => '_e_("==============================
184         ==========\n>>".$thisLogkey."<<")',
185         aa_REVIEW_TYPE          => '_e_$EXCEPTION_TYPE',
186         ab_EXCEPT_FLAGS         => '_e_$eventRequiresFUP',
187         b_RID                   => '_e_$thisRecordID',
188         u_FROM_ADDRESS          => '_e_("<<".$thisFromAddress.">>")',
189         v_SUBJECT               => '_e_("||".$thisSubject."||")',
190         w_BODY_BELOW            => '_e_$thisFullNumberedBody',          # full body
191
192         );
193
194         ## activityLogFile
195
196         %activityLogRecord = (
197         a_LID                   => '_e_$thisLogkey',
198         b_user                  => '_e_$guser."#"/.$Gcfpath',
199         c_datestamp             => '_e_&getDatestamp(\'WIN\',$LGDEBUG_FTN)',
200         e_RID                   => '_e_$thisRecordID',
201         f_clientName            => '_e_$Gclientname',
202         q_eventType             => '_e_$eventType',
```

```perl
203
204    );                        h_eventDesc     => '_e_$eventDesc',
205                              z_FUP_REQUIRED  => '_e_$eventRequiresFUP',
206
207
208
209
210    &getKeyPos_outfile(\%GoutFileRecord);            # identify the fieldwise position of fields in the outfile for canonicalization (use EXC file as model)
211    ############################## DATA ##############################
212    ##print "--- debug --- FRM_ADD FIELDPOS= $fromAddressFieldPos\n";
213
214    $GDSN = 'dsnOrderProcessing';
215
216    if ($Gplatform ne 'WIN') {$relativePath = '/logs/'} else { $relativePath = '\\logs\\';}
217
218    $GactivityLogFileName     = $basePath.$relativePath.'activityLog.tab';
219    $GactivityLogkeysFileName = $basePath.$relativePath.'activityLogkeys.tab';
220
221
222    if ($Gplatform ne 'WIN') {$relativePath = '/db/'} else { $relativePath = '\\db\\';}
223
224    $GexcReportDBkeysFileName      = $basePath.$relativePath.'excReportDBkeys.tab';
225    $GexceptionsDBkeysFileName     = $basePath.$relativePath.'excDetailDBkeys.tab';
226    $GexceptionsBODYDBkeysFileName = $basePath.$relativePath.'excBodyDBkeys.tab';
227    $GoutFileDBkeysFileName        = $basePath.$relativePath.'outFileDBkeys.tab';
228    $GreportFileDBkeysFileName     = $basePath.$relativePath.'reportFileDBkeys.tab';
229
230    if ($Gplatform ne 'WIN') {$relativePath = 'cf'.'/'.$gcfpath} else { $relativePath = '\\cf\\'.$gcfpath\\';}
231
232    $GbodyDataLayoutFileName      = $basePath.$relativePath.'messageBodyDataLayout.cf';
233    $GbodyPreprocRulesFileName    = $basePath.$relativePath.'messageBodyPreprocRules.cf';
234    $GbodyBuyAllFileName          = $basePath.$relativePath.'messageBodyIndicateBuyAll.cf';
235    $GbodyAddressChangeFileName   = $basePath.$relativePath.'messageBodyIndicateAddressChange.cf';
236    $GmailingDBLayoutFileName     = $basePath.$relativePath.'mailingDBLayout.cf';
237
238    ############################## MAIN ##############################
239    ################################################################
240    ################################################################
241
242    $MODE          = '';
243    $MULTI_BUY_ON  = 0;
244    $ADD_CHANGE_ON = 0;
245    $Gsubj_buyToken = uc($Gsubj_buyToken);
246
247    $MODE = "<TOKEN = |$Gsubj_buyToken|>";
248    if($Gsubj_buyToken =~ /\/MULTI/)     { $MULTI_BUY_ON = 1; $MODE .= '<MULTI BUY ON>';}
249    if($Gsubj_buyToken =~ /\/ADDCHANGE/) { $ADD_CHANGE_ON = 1; $MODE .= '<ADDCHANGE ON>'; }
250
251    print "[procorder R.$grevision] $MODE\n".($GDEBUG?"\n<DEBUG: $GDEBUG:$LGDEBUG_FTN>\n\n":"\n");
252
253    ## open CF files
254
255    print "** Opening: $GbodyPreprocRulesFileName...\n";
256    @GPreprocRules = &loadFile_SCALAR($GbodyPreprocRulesFileName);
257
258    print "** Opening: $GbodyBuyAllFileName...\n";
259    @GBuyAllIndicators = &loadFile_SCALAR($GbodyBuyAllFileName);
260
261    print "** Opening: $GbodyAddressChangeFileName...\n";
262    @GAddressChangeIndicators = &loadFile_SCALAR($GbodyAddressChangeFileName);
263
264    print "** Opening: $GbodyDataLayoutFileName...\n";
265    %LAYOUT = &HASHloadFieldMap($GbodyDataLayoutFileName,$LGDEBUG_FTN);
266
267    unless($ADD_CHANGE_ON) {
268    print "** Opening: $GmailingDBLayoutFileName...\n";
269    %MDBLAYOUT = &HASHloadFieldMap($GmailingDBLayoutFileName,$LGDEBUG_FTN);
270
271
```

```perl
272 }
273
274 ## open I/O
275
276 if (!open(FH_log, ">$GactivityLogFilename")) { $eventType = 'ABORT'; $eventDesc = "MAIN::Cannot open GactivityLogFilename: \[$filename\]"; &logE
vent(\)*FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);}
277 if (!open(FH_out, ">$GoutFile")) { $eventType = 'FAIL'; $eventDesc = "MAIN::Cannot open GoutFile: \[$Goutfile\]"; &logEvent(\)*FH_log
$eventType, $eventDesc, $guser, $LGDEBUG_FTN);}
278 if (!open(FH_report, ">$GreportFile")) { $eventType = 'FAIL'; $eventDesc = "MAIN::Cannot open GreportFile: \[$Greportfile\]"; &logEve
FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);}
279 if (!open(FH_excReport, ">$GexcReportFile")) { $eventType = 'FAIL'; $eventDesc = "MAIN::Cannot open excReportFile: \[$GexcReportFile\]"; &logEve
nt(\)*FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);}
280 if (!open(FH_excBODYReport, ">$GexcBODYReportFile")) { $eventType = 'FAIL'; $eventDesc = "MAIN::Cannot open excBODYReportFile: \[$GexcBODYReportFile\]";
&logEvent(\)*FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);}
281 if (!open(FH_exc, ">$GexceptionsFile")) { $eventType = 'FAIL'; $eventDesc = "MAIN::Cannot open GexceptionsFile: \[$GexceptionsFile\]"; &log
Event(\)*FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);}
282
283 ## load %results fieldname keys from layout cf file (now in %LAYOUT)
284
285 @fieldkeys      = undef;
286 @fieldkeys      = &getfieldkeys(\%LAYOUT);
287
288 ##print "--debug--\n"; foreach(@fieldkeys) { print "**".$_."|\n"; }
289
290 $numberRequiredFields   = scalar(@fieldkeys);
291
292 if(!defined($LAYOUT{'orderprocessingDSN'})) { $LAYOUT{'orderprocessingDSN'} = $GDSN; }
293
294 print "** Loading daily order data source: $LAYOUT{'orderprocessingDSN'}...\n";
295 @DB = &read_DSNtoAOH($LAYOUT{'orderProcessingDSN'},$LGDEBUG_FTN);
296
297 $totalRecords   = $#DB +1;
298 $GclientName    = $LAYOUT{'thisclientName'};
299
300 ## startup OK event
301
302 $eventType = '_x_INFO'; $eventDesc = "procOrder R.$Grevision START OK - USER: $guser/$GcfPath CLIENT: $GclientName opDSN ($totalRecords): $LAYOUT{'orderp
rocessingDSN'} mdbDSN: $LAYOUT{'orderProcessingDSN'} -> $LAYOUT{'thismailingTable'} ($totalRecords recs) BODYdefn: $GbodyDataLayoutFilename"; &logEvent(\)
*FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);
303
304 print "\n\n**  Status...\n\n";
305 print "USER:\t\t$guser \n";
306 print "CLIENT:\t\t$GclientName\n";
307 print "cfPATH:\t\t$GcfPath\n\n";
308 print "TOTAL RECORDS \t\t -> \"$LAYOUT{'orderProcessingDSN'}\"...\n";
309 unless($ADD_CHANGE_ON) { print "MAILING TABLE SOURCE \t -> \"$LAYOUT{'thismailingDSN'}\"\n\n"; }
310 print "\n(Note: if this information is not correct, hit <CNTRL> - C to abort and correct <you have 5 seconds>\n\n"; sleep 5;
311
312 print "** Begin processing...\n\n";
313
314 ## flag duplicate e-mail addresses in order stream
315
316 %dupsMap = &checkDupsReturnMap(\@DB, $GDEBUG);
317
318 #################
319 #### main loop ####
320 #################
321
322 $excpCount      = 0;
323 $warnCount      = 0;
324 $outFileCount   = 0;
325 $exceptionsFileCount    = 0;
326 $excReportFileCount     = 0;
327 $reportFileCount    = 0;
328 $FUPcount       = 0;
329 $reportLinecount    = 0;
330
331 for $mainCount (0 .. $#DB) {
332
```

```perl
333 $thisRecordID                               = $mainCount+1;
334 $eventRequiresFUP                           = '';
335 $RECORD_VALID                               = 1;
336 $thisBuyStatus                              = '';
337 $thisShippingAddressBlock                   = undef;
338 $thisBillingAddressBlock                    = undef;
339 $thisMDBBillingAddressBlock                 = undef;
340 $thisFullNumberedBody                       = "\n";
341
342 ## get relevant fields, normalized, this record
343
344 $thisFromAddress   = &normalizespacing(lc($DB[$mainCount]{'FromAddress'}));
345 $thisFromAddress   =~ s/[\t\r]/$Gspace/g;
346
347 $thisFromName      = &normalizespacing($DB[$mainCount]{'FromName'});
348 $thisFromName      =~ s/[\t\r]/$Gspace/g;
349
350 $thisSubject       = &normalizespacing(uc($DB[$mainCount]{'subject'}));
351 $thisSubject       =~ s/[\t\r]/$Gspace/g;
352
353 $thisBody          = uc($DB[$mainCount]{'Body'});
354 $thisBody          =~ s/[\t\r]/$Gspace/g;
355
356 ## split BODY into discrete lines
357
358 @BODYlines   = undef;
359 @topLines    = undef;
360 @bottomLines = undef;
361
362 push(@BODYlines, $1) while ($thisBody =~ s/^([^\012\015]*)(\012\015?|\015\012?)//);
363
364 ## skip lines that are the quoted BODY unless they're the order form -- example delimiters:
365 ## $BODYstartToken = 'HARVARD BUSINESS SCHOOL PUBLISHING';
366 ## $BODYendToken = 'FIRST NAME\:';
367
368 $BODYstartToken   = $LAYOUT{'startDelimiter'};
369 $BODYendToken     = $LAYOUT{'endDelimiter'};
370 $BODYstartToken  =~ s/[\t\n\r]//g;
371 $BODYendToken    =~ s/[\t\n\r]//g;
372 $startToken_ON   = 0;
373 $endToken_ON     = 0;
374 $thisBody        = undef;
375 $thisBodyTop     = undef;
376 $thisBodyBottom  = undef;
377 $lastLineExcluded = undef;
378
379 for($count = 0; $count <= $#BODYlines; $count++) {
380     $line              = $BODYlines[$count];
381     $thisFullNumberedBody .= ('['.$count.'] '.$line."\n");
382
383     unless($startToken_ON || $line =~ /^\s+/) {unless ($line !~ /\S+?/) {$thisBodyTop .= ($line.$Gpipe)}; push(@topLines,('['.$count.'] '.$line.'|')
384
385         if(($line =~ /$BODYstartToken/) .. ($line =~ /$BODYendToken/)) {
386             if(!$startToken_ON && ($line =~ /$BODYstartToken/))    {$startToken_ON = 1;}
387             if(!$endToken_ON && ($line =~ /$BODYendToken/))        {$endToken_ON = 1; $thisBody .= ($line.$Gnewline); unless($line =~ /^\s+/
388
389 || $line !~ /\s+?/ || $line =~ /[\[\]]+/) {$thisBodyBottom .= ($line.$Gpipe)} }
390             next;
391         }
392
393 ## ignore bottom lines if part of the original -> indicators:   ">" || "[" || "]"
394
395 if($startToken_ON && $endToken_ON) { unless($line =~ /^\>+/ || $line =~ /[\[\]]+/) {$thisBodyBottom .= ($line.$Gpipe); push(@bo
396 ttomLines,('['.$count.'] '.$line.'|'));}}
397 $thisBody .= ($line.$Gnewline);
```

```perl
399
400      $thisBodyTop    =~ s/[ \t\r]+//g;
401      $thisBodyBottom =~ s/[ \t\r]+//g;
402
403      if($GDEBUG) { print "\n\n\n** DEBUG__THISBODY=".$thisBody."\n\n\n";}
404
405      ## log/print warning if cannot find FOOTER ID BLOCK: CUSTNO|LISTID|MAILINGID|DIALOGID || if bad match yields the same for CUSTNO|LISTID
406
407          $thisCustNo          = -1;
408          $thisListID          = -1;
409          $thisMailingID       = -1;
410          $thisDialogID        = -1;
411          $LISTCID_EXCEPT_ON   = 0;
412          $MAILINGID_EXCEPT_ON = 0;
413
414      if($thisBody !~ /\[\]\([^\]]*?\)\|\([^\]]*?\)\|?\([^\]]*?\)\|\]/) {
415
416          $LISTCID_EXCEPT_ON = 1;
417          $eventType = 'EX_ERR'; $excpCount++; $eventDesc = " \<".$thisRecordID."\>   UNMATCHED FOOTER ID BLOCK: CUSTNO|LISTID|MAILINGID|DIALOGID"; &logEvent(
418          \*FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);
419          $eventType = 'INFO'; $eventDesc = " \<".$thisRecordID."\>   >>> ATTEMPTING SUBJECT LINE MAILING ID RECOVERY >>>"; &logEvent(
420          \*FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);
421          ## test to see if there is a cf file synonym (**,++, etc) for the MAILING ID which can be recovered in the absence of the footer token
422
423          if($thisMailingID == -1) {  $thisMailingID = &subjectSynonymLookup($thissubject,\%LAYOUT); }
424
425          if($thisMailingID == -1) {
426              $MAILINGID_EXCEPT_ON = 1;
427              $RECORD_VALID=0;
428              $eventType = 'EX_ERR'; $excpCount++; $eventDesc = " \<".$thisRecordID."\>   MAILING ID RECOVERY WAS NOT SUCCESSFUL"; &logEvent(\*F
429          H_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN); }
430
431          else {
432
433
434              $eventType = 'INFO'; $excpCount++; $eventDesc = " \<".$thisRecordID."\>   <<< MAILING ID RECOVERY SUCCESSFUL USING: $thissubjectMa
435          ilingIDToken -> $LAYOUT{$thissubjectMailingIDToken}"; &logEvent(\*FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);
436          }
437      }
438
439      else {
440          $element1 = $1; $element2 = $2; $element3 = $3; $element4 = $4;
441          if($GDEBUG) { print "****DEBUG_   element1= $1 ___ element2= $2 ___ element3= $3 ___ element4= $4\n"; sleep 1; }
442
443
444
445
446          $thisCustNo     = ($element1 =~ /\S+?/) ? $element1 : -1;
447          $thisListID     = ($element2 =~ /\S+?/) ? $element2 : -1;
448          $thisMailingID  = ($element3 =~ /\S+?/) ? $element3 : -1;
449          $thisDialogID   = ($element4 =~ /\S+?/) ? $element4 : -1;
450
451          if ($thisCustNo == -1 && $thisListID == -1) {
452              $LISTCID_EXCEPT_ON = 1;
453              $MAILINGID_EXCEPT_ON = 1;
454              $eventType = 'EX_ERR'; $excpCount++; $eventDesc = " \<".$thisRecordID."\>   BAD MATCH: CUSTNO, LISTID"; &logEvent(\*FH_log, $even
455          tType, $eventDesc, $guser, $LGDEBUG_FTN);
456          }
457          ## log/print INFO if cannot find mailingID
458          if($thisMailingID == -1) { $eventType = 'INFO'; $eventDesc = " \<".$thisRecordID."\>   NO MAILING ID MATCH"; &logEvent(\*FH_log, $eventTyp
459          e, $eventDesc, $guser, $LGDEBUG_FTN);}
460
461          ## log/print INFO if cannot find dialogID
```

```
462 .
463     if($thisDialogID == -1) { $eventType = 'INFO'; $eventDesc = " \<".$thisRecordID."\>     NO DIALOG ID MATCH"; &logEvent(\*FH_log, $eventType,
464     $eventDesc, $Guser, $LGDEBUG_FTN);}
        }

466 ## check for /multi || quoted BUY TOKEN (eg: "YES")
467
468     $buyTokenPosition            = 'none';
469     $thisItemsSelectedstack      = undef;
470     @thisItemsSelected           = undef;
471
472     unless($ADD_CHANGE_ON) {
473
474     ## check for buy tokens, eg: YES in SUBJECT, TOP, BOTTOM || letter items: ABC || A,B,C || A-C
475
476     $buyTokenPosition = &detectBuyTokens($MULTI_BUY_ON,$Gsubj_buyToken,$thisSubject,$thisMailingID,$thisBodyTop,$thisBodyBottom,\%LAYO
UT,\@topLines,\@bottomLines,$GDEBUG);
477
478     ## check for no-match case and except
479
480     if($buyTokenPosition eq 'none') { $eventRequiresFUP .= '.noTok' ; $FUPcount++; $eventType = 'ERR_FUP'; $warnCount++;     $eventDesc =
" \<".$thisRecordID."\>     SUBJECT, TOP, BOTTOM -> NO BUY TOKEN MATCH ($Gsubj_buyToken)"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser,     $LGDEBUG_FTN)
; }

481     }
482
483     ## if match and MULTI_BUY_ON, prepare order items listing
484
485     if($buyTokenPosition ne 'none' && $MULTI_BUY_ON) {
486
487     $thisItemsSelectedstack = &extractDedupItemsselectedstack(@thisItemsSelected);
488 .
489     }
490
491     ####################
492     ## INNER LOOP ##
493     ####################
494
495     %results              = {};
496     $countFields          = 0;
497     $nullCriticalFields   = 0;
498     $ADDRESS_ADEQUATE     = 1;
499
500 ## process body fields by key; keys == standard field names; values == this BODY's field names mapped thereon
501
502     foreach (@fieldkeys) {
503
504     $key = $_;
505
506     ## print "--DEBUG-- key = |$key| \n";
507
508     $thisLayoutFieldName = uc($LAYOUT{$key});
509     $thisLayoutFieldName =~ s/\s+$//;
510
511     ###################################
512     ###### MAIN MATCHING BLOCK ########
513     ###################################
514
515     $MATCH_DATA_ON = 0;
516     $RULE = -1;
517
518     $expectedMatch      = $thisLayoutFieldName.'\:.*?\[([^\]]+?)\]';
519     $noBracketsMatch    = $thisLayoutFieldName.'\:.*?[^\A\012\015]+?)';
520
521     CASE: {
522
523     TRY_EXPECTED_MATCH:      if($thisBody =~ /$expectedMatch/)      { $MATCH_DATA_ON = 1; $matchElement = $1; $RULE = 1; last
524 CASE: }
        TRY_NO_BRACKETS_MATCH:   if($thisBody =~ /$noBracketsMatch/)    { $MATCH_DATA_ON = 1; $matcheElement = $1; $RULE = 2; last
        CASE: }
```

```perl
525
526         }
527
528    ## A MATCH WAS FOUND, THIS KEY ... push into results set
529
530        if($MATCH_DATA_ON) {
531
532            $countFields++;
533            $matchElement =~ s/[^\w\s\-\.\/\\#\!\|\@]/$Gspace/g;
534            $results{$key} =  &normalizeSpacing($matchElement);
535
536    ## --DEBUG-- result = |$results{$key}| \n";
537
538    ## build shipping and billing address blocks
539
540
541        CASE: {
542
543            if($key =~ /u[a-z]_/) {
544
545            ## construct SHIPPING ADDRESSBLOCK for report
546            if ($key =~ /[hijklmq]\_/) { $thisshippingAddressBlock .= (length($results{$key}) > 0)?($results{$key}.$Gpipe):'';
547            if ($key =~ /[efgnop]\_/) { $thisshippingAddressBlock .= (length($results{$key}) > 0)?($results{$key}.$Gspace):'';
548
549            last CASE;
550            }
551
552            if($key =~ /[^u]_/) {
553
554            ## construct BILLING ADDRESSBLOCK for report
555            if ($key =~ /[hijklmq]\_/) { $thisBillingAddressBlock .= (length($results{$key}) > 0)?($results{$key}.$Gpipe):'';
556            if ($key =~ /[efgnop]\_/) { $thisBillingAddressBlock .= (length($results{$key}) > 0)?($results{$key}.$Gspace):'';
557
558            last CASE;
559            }
560            }
561
562
563    ## check for null critical fields: any absence here will throw exception, unless CUSTNO
564
565    ## print "--debug-- |$LAYOUT{'criticalFields'}| \n";sleep 1;
566
567        if ($key =~ /[$LAYOUT{'criticalFields'}]\_/ && $results{$key} !~ /\s+?/ && $thisCustNo < 1) {
568
569            $RECORD_VALID=0; $ADDRESS_ADEQUATE = 0;
570            $eventRequiresFUP .= (($eventRequiresFUP =~ /badAd/)?'':'; badAd'); $FUPcount++; $eventType = 'EX_FUP'; $excpCount++; $even
571            tDesc = " \<\.".$thisRecordID."\>                            CRITICAL STANDARD FIELD.VALUE IS NULL: \[$thisLayoutFieldName\]"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $L
572            GDEBUG_FTN);
573
574        }
575
576    ## ALL MATCHES FAILED, THIS KEY
577
578        if(!$MATCH_DATA_ON) {
579
580            $results{$key} = undef;
581            $eventType = 'EX_ERR'; $excpCount++; $eventDesc = " \<\.".$thisRecordID."\>          STANDARD FIELD_NAME=FIELD_VALUE PAIR UNPARSABLE: \[$th
582            isLayoutFieldName\]"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
583
584            # check to see is unparsable field is a critical field
585            # critical fields -> from CF file: any absence here will throw exception
586
```

```
587     if ($key =~ /[$LAYOUT{'criticalfields'}]\_/) {
588
589                 $RECORD_VALID=0; $ADDRESS_ADEQUATE = 0;
590                 $eventRequiresFUP .= (($eventRequiresFUP =~ /badad/)?'':'.badad'); $FUPcount++; $eventType = 'EX_FUP'; $excpcount++; $even
591                 $eventDesc = " \<".$thisRecordID."\>  UNPARSABLE STANDARD FIELD_NAME=FIELD_VALUE PAIR IS CRITICAL: \[$thisLayoutFieldname]"; &logEvent(\*FH_log, $eventType,
592     $eventDesc, $guser, $LGDEBUG_FTN);
593             }
594         }
595
596     ############################
597     #### END INNER LOOP ###
598     ############################
599     ############################
600     ############################
601     ### SUMMARY RULES ###
602     ############################
603     ############################
604
605     ## throw EX_ LISTNO/CUSTNO error exception if address is not OK && !phone, depending on CID_LID_REQUIRED from cf file
606
607         if($LAYOUT{'CID_LID_REQUIRED'}) {
608
609             if($LISTCID_EXCEPT_ON) {
610
611                 $RECORD_VALID = 0;
612 c = " \<".$thisRecordID."\>  $eventRequiresFUP .= (($eventRequiresFUP =~ /chkcLID/)?'':'.chkcLID'); $FUPcount++; $eventType = 'EX_FUP'; $excpcount++; $eventDes
613                 EITHER CID || LID ABSENT WHEN CF CRITICAL"; &logEvent(\*FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);
614             }
615         }
616     if(!$LAYOUT{'CID_LID_REQUIRED'}) {
617
618             if($LISTCID_EXCEPT_ON && !($ADDRESS_ADEQUATE && $results{'r\_phone'} !~ /\s+?/)) { $RECORD_VALID = 0; }
619
620         }
621
622
623     ## throw EX_ MAILINGID exception if cf file says it's required
624
625         if($LAYOUT{'MAILINGID_REQUIRED'}) {
626
627             if($MAILINGID_EXCEPT_ON) {
628
629                 $RECORD_VALID = 0;
630                 $eventRequiresFUP .= (($eventRequiresFUP =~ /chkMID/)?'':'.chkMID'); $FUPcount++; $eventType = 'EX_FUP'; $excpcount++; $eventDesc
631 = " \<".$thisRecordID."\>  MAILING ID ABSENT WHEN CF CRITICAL"; &logEvent(\*FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);
632             }
633         }
634
635
636     ## throw INFO if less than number required
637
638         if(($countFields < $numberRequiredFields) && $thisCustno < 1) {
639             $eventType = 'INFO'; $eventDesc = " \<".$thisRecordID."\>  (OVERALL) NOT ALL STANDARD FIELDS FOUND: $countFields ($numberRequiredfields)"
640     ; &logEvent(\*FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);
641         }
642
643
644     ## tag duplicates for FUP
645     if($dupsMap{$mainCount} =~ /\/#\/#.+\/#\/#/) { $eventRequiresFUP .= (($eventRequiresFUP =~ /dupAd/)?'':'.dupAd'); $FUPcount++; }
646
647     ## catch thisBodyTop and thisBodyBottom blank yet address still extracted adequately (eg happens when cut & paste responses between the body delimiters)
648
649     if($RECORD_VALID && $thisBodyTop !~ /\S+?/ && $thisBodyBottom !~ /\S+?/) {
650     if($RECORD_VALID && $thisBodyTop !~ /\S+?/ && $thisBodyBottom !~ /\S+?/) {
```

```perl
651          $RECORD_VALID = 0;
652          $eventType = 'EX_ERR' ; $eventRequiresFUP .= '.chkBody' ; $FUPcount++; $eventDesc = " \<".$thisRecordID."\>   CUSTOMER RESPONSE (Top,
             Bottom) NOT FOUND WHEN EXPECTED"; &logEvent(\*FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);
653          }
654
655     ###################################################################
656     ###################################################################
657     #### RECORD INVALID: Attempt recovery from mailing database ####
658     ###################################################################
659     ###################################################################
660     $RECORD_RECOVERED = 0;
661
662     if(!$RECORD_VALID && !$ADD_CHANGE_ON) {
663
664          $eventType = 'INFO'; $eventDesc = " \<".$thisRecordID."\>   >>> BAD RECORD: ATTEMPTING RECOVERY-LOOKUP USING \"$thisFromAddress\", $LAYOUT{'thisma
             ilingDSN'} -> $LAYOUT{'thisMailingTable'}"; &logEvent(\*FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);
665
666          %recoveredResults = undef;
667          %recoveredResults = &querySELECT_DB(\%LAYOUT, 'EMAIL', $thisFromAddress,$LGDEBUG_FTN);
668
669     ## foreach(keys(%recoveredResults)) { print "=debug== key = $_       value = $recoveredResults{$_} \n"; }
670
671          CASE: {
672
673
674          if($recoveredResults{'_exit_'} < 1) {
675                $RECORD_RECOVERED = 0;
676                $eventType = 'EX_ERR'; $eventRequiresFUP .= (($eventRequiresFUP =~ /goodMDB/)?'':'.goodMDB'); $FUPcount++; $excpCount++; $eventDesc
                  = " \<".$thisRecordID."\>   >>> MAILING TABLE RECOVERY-LOOKUP FAILED ON \"$thisFromAddress\""; &logEvent(\*FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_F
                  TN);
677
678                last CASE;
679          }
680
681          if($recoveredResults{'_exit_'} == 1) {
682
683                $RECORD_RECOVERED = 1;
684                $RECORD_VALID = 1;
685                $eventType = 'WRN_FUP'; $eventRequiresFUP .= (($eventRequiresFUP =~ /badMDB/)?'':'.badMDB') ; $FUPcount++; $excpCount++; $eventDe
                  sc = " \<".$thisRecordID."\>   <<< MAILING TABLE RECOVERY-LOOKUP SUCCESSFUL USING \"$thisFromAddress\""; &logEvent(\*FH_log, $eventType, $eventDesc, $guse
                  r, $LGDEBUG_FTN);
686
687          ## construct $thisMDBbillingAddress
688
689                foreach $key(sort(keys(%MDBLAYOUT))) {
690
691                      $value = $MDBLAYOUT{$key};
692
693          ## print "**DEBUG key = $key ___ value = $value \n";
694
695                      if ($key =~ /[hijklmq]\/) { $thisMDBbillingAddressBlock .= (length($recoveredResults{$value}) > 0)?($recoveredResults{$va
                       lue}.$Gpipe):''; }
696                      if ($key =~ /[efgnop]\/) { $thisMDBbillingAddressBlock .= (length($recoveredResults{$value}) > 0)?($recoveredResults{$val
                       ue}.$Gspace):''; }
697
698                }
699
700          ## restore mailingID and CID, if blank
701
702          ## print "**DEBUG mailingID = $recoveredResults{'ED_MAILINGID'} ___ custno = $recoveredResults{'CID'} ___ current custno = |$thisCustNo \n";
703                if($thisMailingID !~ /\s+?/ || $thisMailingID == -1)        { $thisMailingID        = $recoveredResults{'ED_MAILINGID'}; }
704                if($thisCustNo !~ /\S+?/ || $thisCustNo == -1)              { $thisCustNo           = $recoveredResults{'CID'}; }
705
706                last CASE;
707
708          }
709
710          }
711     }
```

```
712  }
713
714  #### add an empty flag where SHIPPING ADDRESS is returned blank
715
716  if($thisshippingAddressBlock !~ /\S+?/) { $thisshippingAddressBlock = '{ SAME }'; }
717  if($thisshippingAddressBlock !~ /\S+?/) { $thismDBbillingAddressBlock = '{ N/A }'; }
718  if($thismDBbillingAddressBlock !~ /\S+?/) { $thismDBbillingAddressBlock = '{ N/A }'; }
719
720  #### tag address block exceptions with diagnostic ques
721
722  if(!$RECORD_VALID) {
723
724
725      CASE: {
726
727          if($thisBillingAddressBlock !~ /\S+?/ && ($thisBodyTop =~ /\S+?/ || $thisBodyBottom =~ /\S+?/))
728          { $thisBillingAddressBlock = '{ =NO_
729          DY_EMPTY/INSPECT'; last CASE;}
730          if($thisBillingAddressBlock !~ /\S+?/ && $thisBodyTop !~ /\S+?/ && $thisBodyBottom !~ /\S+?/)
731          { $thisBillingAddressBlock = '{ BO
732          _FLD_NUL/INSPECT'; last CASE; }
733          if($thisBillingAddressBlock =~ /\S+?/ && !$ADDRESS_ADEQUATE)
734          { $thisBillingAddressBlock = '{ CRIT
735          ADDRESS_ADEQUATE)
736
737          $thisBillingAddressBlock = '=UNKNOWN_ERR/INSPECT=|'.$thisBillingAddressBlock; last CASE;
738
739      ###############################################
740      ## write valid record data to outfiles ##
741      ###############################################
742
743  $eventRequiresFUP = ($eventRequiresFUP !~ /\S+?/) ? '.inspect' : $eventRequiresFUP;
744
745  }
746
747  $ED_ORDER_VALUE = 1;
748  $EXCEPTION_TYPE = 'ok';
749  $UPDATE_SUCCESSFUL = 0;
750  $THIS_KEY_FIELD = undef;
751  $THIS_KEY_FIELD_VALUE = undef;
752
753      if($RECORD_VALID) {
754
755          unless($ADD_CHANGE_ON) {
756
757          CASE: {
758
759              if(&queryUPDATE_DB(\%LAYOUT,$THIS_KEY_FIELD_VALUE = $thisFromaddress,$THIS_KEY_FIELD = "EMAIL","ED_ORDER",$ED_ORDER_VALUE,
760              { $UPDATE_SUCCESSFUL=1; last CASE; }
761              if(&queryUPDATE_DB(\%LAYOUT,$THIS_KEY_FIELD_VALUE = $thisCustNo,$THIS_KEY_FIELD = "CID","ED_ORDER",$ED_ORDER_VALUE,"ED_
762              { $UPDATE_SUCCESSFUL=1; last CASE; }
763              if(&queryUPDATE_DB(\%LAYOUT,$THIS_KEY_FIELD_VALUE = $thisListID,$THIS_KEY_FIELD = "ED_LID","ED_ORDER",$ED_ORDER_VALUE,"ED_
764  MAILACTION",-1,$RECORD_RECOVERED,$LGDEBUG_FTN);
765
766  "ED_MAILACTION",-1,$RECORD_RECOVERED,$LGDEBUG_FTN)) { $UPDATE_SUCCESSFUL=1; last CASE; }
767  LACTION",-1,$RECORD_RECOVERED,$LGDEBUG_FTN);
768  MAILACTION",-1,$RECORD_RECOVERED,$LGDEBUG_FTN);
769              if(!$UPDATE_SUCCESSFUL) {
770
771                  $RECORD_VALID = 0;
```

```
772
773        }
774
775 #### Buy tokens not found and not ADDRESS_CHANGE; move to EXC report
776     unless($ADDRESS_CHANGE) { if($buyTokenPosition eq 'none') { $RECORD_VALID = 0; $ED_ORDER_VALUE = 2; } }
777     if($RECORD_VALID) {
778
779
780        $reportlineCount++;
781
782
783        ## update MAILING TABLE
784
785
786        ## write out files
787
788        &writeRecord(\*FH_out,$gfielddelimiter,($thiskey=&getNextDBkey_return($GoutFiledBkeysFilename,$LGDEBUG_FTN)),\%GoutFileRecord,'EXTENDED',
789     RMAL',$LGDEBUG_FTN);
         &writeRecord(\*FH_report,$gfielddelimiter,($thiskey=&getNextDBkey_return($GreportFiledBkeysFilename,$LGDEBUG_FTN)),\%GreportFileRecord,'NO
$LGDEBUG_FTN);
790
791        ## include ALL BODYs in the EXC_BODY file for potential review
792
793        if($eventRequiresFUP =~ /\S+?/) { $EXCEPTION_TYPE = 'SOFT-EXC'; }
794
795              &writeRecord(\*FH_excBODYReport,$gfielddelimiter,$thisBODYlogkey=&getNextDBkey_return($GexceptionsBODYDBkeysFilename,$LGDEBUG_FTN)
,\%GexceptionsBODYFileRecord,'BODY',$LGDEBUG_FTN);
796
797        $outFileCount++;$reportFileCount++;
798        }
799 ################################################
800 ################################################
801 ## write exception record data to outfiles ##
802 ################################################
803 ################################################
804
805
806        if(!$RECORD_VALID) {
807
808        $EXCEPTION_TYPE = 'HARD-EXC';
809        &writeRecord(\*FH_exc,$gfielddelimiter,$thiskey=&getNextDBkey_return($GexceptionsDBkeysFilename,$LGDEBUG_FTN),\%GoutFileRecord,'EXTENDED',
810     RMAL',$LGDEBUG_FTN);
         &writeRecord(\*FH_excReport,$gfielddelimiter,$thiskey=&getNextDBkey_return($GexcReportDBkeysFilename,$LGDEBUG_FTN),\%GreportFileRecord,'NO
811        &writeRecord(\*FH_excBODYReport,$gfielddelimiter,$thisBODYlogkey=&getNextDBkey_return($GexceptionsBODYDBkeysFilename,$LGDEBUG_FTN),\%Gexce
ptionsBODYFileRecord,'BODY',$LGDEBUG_FTN);
812
813        $exceptionsFileCount++; $excreportFileCount++;
814        }
815        print "\n";
816
817
818
819 ################################
820 #### end main loop ####
821 ################################
822
823 ## print STDOUT exit state
824
825 $dateNow = &getDateStamp('WIN',$LGDEBUG_FTN);
826
827 print "\n** [procOrder version $grevision] Done!   $totalRecords records processed by $Guser/$gcfpath for client: $gclientName **\n\n";
828 print "SUMMARY REPORT for: ".$dateNow."\n\n";
829 print "FILES WRITTEN \n";
830 printf "\n%10d",$reportFileCount;
831        print "  records written to REPORT FILE: $greportFile";
832 printf "\n%10d",$excReportFileCount;
833        print "  records written to EXC_REPORT FILE: $gexcReportFile";
834 printf "\n%10d",$outFileCount;
```

```perl
835 print      "  records written to OUTPUT FILE (all fields): $goutFile";
836 printf     "\n%10d", $exceptionsFileCount;
837 print      "  records written to EXCEPTIONS FILE (all fields): $gexceptionsFile";
838 print      "\n\nLOGGING\n\n";
839 print      "All rule failure events for all time logged in detail to: $gactivityLogFileName\n";
840 printf     "\n%10d", $totalRecords;
841 print      "  BODYs for this session are keyed, categorized in: $GexcBODYReportFile, for review in an editor\n";
842 print
843 $eventType = '_x_INFO'; $eventDesc = "procOrder R.$grevision EXIT OK - USER: $guser/$gcfpath CLIENT: $gclientName     opDSN ($totalRecords): $LAYOUT{'orderPr
    ocessingDSN'} mbdDSN: $LAYOUT{'orderProcessingDSN'}-> $LAYOUT{'thisMailingTable'} ($totalRecords recs) BODYdefn: $GbodyDataLayoutFileName"; &logEvent()\*
    FH_log, $eventType, $eventDesc, $eventDesc, $guser, $LGDEBUG_FTN);
844 close(FH_out); close(FH_exc); close(FH_log); close(FH_report); close(FH_excReport);close(FH_excBODYReport);
845 exit;
846
847
848 ################################################
849 ################################################
850 ################# SUBS #########################
851 ################################################
852 ################################################
853 #
854 # ---------------------------------------------
855 #
856 #
857 sub alphaordered {
858
859 my(@singleElementArray) = @_;
860
861 my  $alpha       = 0;
862 my  $i           = 0;
863
864 my  $base = $singleElementArray[0];
865
866 for($i = 1; $i <= $#singleElementArray; $i++) {
867
868     $ordBase = ord($base); $ordArr = ord($singleElementArray[$i]);
869
870 ##  print "==debug=$i= base = |$base| ordBase= $ordBase ____ singleElementArray = $singleElementArray[$i] ordArr = $ordArr \n";
871
872     if($ordBase > $ordArr) { return $alpha; }
873     $base = $singleElementArray[$i];
874
875 }
876
877 $alpha = 1;
878
879 return $alpha;
880 }
881
882
883 #
884 # ---------------------------------------------
885 #
886 sub canonicalize {
887
888 my($inRecord) = @_;
889 my($elementsIndex, $fieldsIndex, $rec, $buf, $field) = undef;
890 my(@field) = undef;
891 my  $space = ' ';
892 my  $dashEscape = $space.'_d_esc_'.$space;
893
894 $inRecord = uc($inRecord);
895
896 my @allFields = undef;
897
898 @allFields = split(/$gfieldDelimiter/,$inRecord);
899 my $fieldsIndex = undef;
900
901 for($fieldsIndex=0; $fieldsIndex <= $#allFields; $fieldsIndex++) {
```

```perl
902
903
904
905   if ($fieldsIndex != $emailFieldPos && $fieldsIndex != $fromAddressFieldPos) {
906       if( $fieldsIndex == $companyFieldPos || $fieldsIndex == $lastNameFieldPos || $firstNameFieldPos || $fieldsIndex == $titl
          eFieldpos || $fieldsIndex == $countryFieldPos || $fieldsIndex == $departmentFieldPos || $fieldsIndex == $cityFieldPos) { $allFields[$fieldsIndex] == s/\
          -/$dashEscape/g; }
907       $allFields[$fieldsIndex]  =~ s/\s+$//;          # trailing space
908       $allFields[$fieldsIndex]  =~ s/^\s+//;          # leading space
909       $allFields[$fieldsIndex]  =~ s/[\s"\;,.\\\/\(\)]/$space/g;   # extraneous chars
910       $allFields[$fieldsIndex]  =~ s/\s{2,}/$space/g;  # more than one separating space
911
912   }
913   elsif ($fieldsIndex == $emailFieldPos || $fieldsIndex != $fromAddressFieldPos) { $allFields[$fieldsIndex] =~ s/$space//g; }
914
915   @field = split($space,$allFields[$fieldsIndex]);
916
917   $field = undef;
918
919   for($elementsIndex = 0; $elementsIndex <= $#field; $elementsIndex++) {
920
921       CONVERT_TITLE_CASE: {
922
923           $leaveupper = 1;
924
925   LE_CASE; }
926
927       if($fieldsIndex == $postalCodeFieldPos) { last CONVERT_TITLE_CASE; }
928       if($headerFields[$fieldsIndex]!~ /\w*Ix\w*$/ && $field[$elementsIndex] =~ /^[AEIOU\.][^AEIOU\.]?$/) { last CONVERT_TIT
929
930       if($field[$elementsIndex] =~ /^[A-Z]\.[A-Z]?\.?$/)         { last CONVERT_TITLE_CASE; }
931       unless($field[$elementsIndex] =~ /^(AND)|(OR)|(NEW)|(UND)|(AT)|(CO)|(LTD)|(INC)|(THE)|(RD)/) {
932   [A-Z]\.[A-Z]?$/))                                              && ($field[$elementsIndex] =~ /^[A-Z]\.?[A-Z]\.?[A-
      z]$/ || $field[$elementsIndex] =~ /^[^J]S\.?[A-Z]\.?$/))   { last CONVERT_TITLE_CASE; }
933           if($fieldsIndex == $companyFieldPos)             && ($field[$elementsIndex] =~ /^[A-Z]\.?[A-
      tsIndex] !~ /[MD][RS]/)                                      =~ /^[ARHMGIVUCS][EMSQ&TRAFIOPV][PADO\.]*$/ && $field[$elemen
934   ~ /^[A-Z][A-Z]\.?$/ )                                     if($fieldsIndex == $countryFieldPos && ($field[$elementsIndex] =
935                                                            =~ /^[A-Z]\.[A-Z]?\.?$/ || $field[$elementsIndex] =
936
937                                                            { last CONVERT_TITLE_CASE; }
938       unless($fieldsIndex == $stateFieldPos && length($field[$elementsIndex])>5) { if($fieldsIndex == $stateFieldpos || $fieldsIndex ==
          $postalCodeFieldpos)       { last CONVERT_TITLE_CASE; } }
939
940           $leaveupper = 0;
941
942           $buf = lc($field[$elementsIndex]);
943
944           if($fieldsIndex != $emailFieldPos);
945
946       }
947
948       if($leaveupper) { $buf = $field[$elementsIndex]; }
949       if($fieldsIndex != $fromAddressFieldPos)      { $buf = ucfirst($buf); last CONVERT_TITLE_CASE; }
950       }
951       $field .= ($buf.$space);
952   }
953   $field .= $field.$fieldDelimiter);
954   }
955
956   chop($rec);
957
958   $rec .= ($field.$fieldDelimiter);
959
960   $rec =~ s/$dashEscape/\-/g;
961   } return $rec;
962   }
```

```perl
963   #
964   #
965   #
966   #
967   sub checkDupsReturnMap {
968
969     my($ptrAOHDB,$GDEBUG)   = @_;
970     my(@DB)                 = @$ptrAOHDB;
971
972     ## prepare map of duplicates on FromAddress
973
974     my %dupsMap             = undef;
975     my $j                   = undef;
976
977     for $j (0 .. $#DB) {
978
979       $dupsMap{$j} = $DB[$j]{'FromAddress'}
980
981     }
982
983     my @mirrorDupsMap       = undef;
984     @mirrorDupsMap          = values(%dupsMap);
985     my $key                 = undef;
986     my $value               = undef;
987     my @arr                 = undef;
988     my $number              = undef;
989
990     foreach $key(keys %dupsMap) {
991
992       $value                = $dupsMap{$key};
993       next if($value !~ /\S+?/);
994
995       @arr                  = undef;
996       @arr                  = grep /$value/, @mirrorDupsMap;
997       $number               = scalar(@arr);
998
999       ## print "** key=".$key." *** value=".$dupsMap{$key}."__occurrences=".$number."\n";
1000
1001      if($number > 1) {
1002
          $eventType = 'INFO'; $eventDesc = "DUPLICATE FromAddress ITEM DETECTED ON IMPORT & FLAGGED: ($number|$key|$value)"; &logEvent(\*FH_lo
1003      g, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);
          $dupsMap{$key} = ( '###'.$number.'### '.$dupsMap{$key});
1004
1005      }
1006
1007      ## print "** key=".$key." *** value=".$dupsMap{$key}."__occurrences=".$number."\n";
1008    }
1009
1010    return %dupsMap
1011  }
1012
1013  #
1014  # -----
1015  #
1016  sub checkMultiBuy_Extract {
1017
1018    my($thisLine,$thismailingID,$pos,$ptrLAYOUT,$GDEBUG) = @_;
1019
1020    my($exit) = 0;
1021
1022    my(%LAYOUT)             = %$ptrLAYOUT;
1023    my(@rangeStack)        = undef;
1024    my(@singleStack)       = undef;
1025    my($countRangeMatches) = 0;
1026    my($repeat)            = 1;
1027    my($match)             = undef;
1028    my($replacementEscChar) = '%';
1029    my($preProcRemoveChar) = '^';
1030    my($elementSeparator)  = ',';
```

```perl
1031  ## PREPARE THE LINE FOR MATCHING
1032
1033  $thisLine                    = uc($thisLine);
1034
1035  ## RUN RULESET FROM RULES CF FILE
1036
1037  foreach(@GpreProcRules) { eval; }
1038
1039  my $thisPreExtractionLine    = $thisLine;
1040  my @testSingleStack          = undef;
1041  my $maxASCII                 = 0;
1042
1043  ## EXTRACTION LOOP
1044
1045  while($repeat) {
1046
1047      my $RULE  = 0;
1048      my $match = undef;
1049
1050  ##print "\n\n--debug-- maxASCII -- thisLine IS CURRENTLY |$thisLine| \n\n";
1051
1052      MATCH_CASE: {
1053
1054          SINGLE_CASE: {
1055
1056  ##
1057              if($thisLine =~ /^([A-z])$/)                                      { $match = $1;  $RULE = -1;  last MATCH_CASE; }
1058              if($thisLine =~ /^([A-z])[^A-z]{1}[^A-z\%\@]{1}/)                { $match = $1;  $RULE = -2;  last MATCH_CASE; }
1059              if($thisLine =~ /[^A-z]([A-z])[^A-z\%\@\-]{1}[A-z\%\@\-]*/)      { $match = $1;  $RULE = -3;  last MATCH_CASE; }
1060          }
1061
1062          RANGE_CASE: {
1063
1064              if($thisLine =~ /^([A-z]\-[A-z])[^A-z\%\@]{1}/)                          { $match = $1;  $RULE = 1;  last MATCH_CASE; }
1065              if($thisLine =~ /[^A-z\%\@]{1}([A-z]\-[A-z])[^A-z]}/)                    { $match = $1;  $RULE = 2;  last MATCH_CASE; }
1066              if($thisLine =~ /([A-z]\-[A-z])*/)                                       { $match = $1;  $RULE = 3;  last MATCH_CASE; }
1067  ##          if($thisLine =~ /[^A-z]{1}([A-z]{1}[A-z]\%\@]{1})/)
1068  ##          if($thisLine =~ /^[A-z]{1}([A-z][A-z]*)/)                               { $match = $1;  $RULE = 4;  last MATCH_CASE; }
1069          }
1070
1071      }
1072
1073      $repeat = 0;
1074      }
1075
1076      $thisLine =~ s/$match/$replacementEscchar/g;
1077
1078      $thisCheckedMailingID = ($thisMailingID!=0) ? $thisMailingID : -1;
1079
1080  ##print "\n-- debug - ELEMENT CONSIDERED with break = $RULE |$match| \n";
1081      if($GDEBUG) { print "--debug-- BEGIN RULES -- RANGE = |$LAYOUT{$thisCheckedMailingID}| from $thisCheckedMailingID \n"; }
1082
1083  ## BEGIN RULES TO SEE WHAT WE GOT ##
1084
1085      $PUSH = 1;
1086      if($RULE > 0) {
1087
1088          my(@arr) = undef;
1089          my $tempMatch = $match;
1090          $tempMatch =~ s/\-//g;
1091          @arr = split(//,$tempMatch);
1092
1093          if(!&alphaordered(@arr)) {
1094
1095  ## print "-- debug -- REJECT - not alpha match |$match| in line |$thisPreExtractionLine| \n";
1096              $PUSH = 0;
1097          }
1098      }
1099  }
```

```perl
1100    }
1101
1102
1103    #if($RULE > 0 && ord($arr[0]) <= $maxASCII) {
1104
1105    #    print "-- debug -- REJECT - range case |$match| out of alpha order with current stack!\n";
1106    #    if(ord($match)>$maxASCII) {$maxAscii = ord($match); }
1107    #    $PUSH = 0;
1108
1109    #}
1110
1111    if(defined($match) && $match !~ /[$LAYOUT{$thisCheckedMailingID}]/) {
1112
1113        ## print "--debug-- REJECT - not in RANGE = |$LAYOUT{$thisCheckedmailingID}| match |$match| in line |$thisPreExtractionLine|  \n" ;
1114        if(ord($match)>$maxASCII) {$maxAscii = ord($match); }
1115        $PUSH = 0;
1116
1117
1118
1119    if($PUSH && $RULE > 0 && $match =~ /\s+?/) {
1120
1121    ##   push(@rangestack, ('<'.$RULE.'> |'.$match)); $exit = 1;
1122        $match =~ s/\-/ thru /g;
1123        push(@rangestack, $match);
1124        $exit=1;
1125
1126
1127    if($PUSH && $RULE < 0 && $match =~ /\s+?/) {
1128
1129        push(@testsinglestack,$match);
1130
1131    unless(ord($match) <= $maxASCII) {
1132
1133    ##       push(@singlestack, ('<'.$RULE.'> |'.$match));
1134             push(@singlestack, $match);
1135             $exit=1;
1136
1137
1138        if(ord($match)>$maxASCII) {$maxAscii = ord($match); }
1139
1140    }
1141    if($GDEBUG) { if(!defined($match)) {print "--debug-- match not defined! in line= |$thisPreExtractionLine|\n"; }}
1142
1143    ## consolidate extractions, pre report
1144
1145    foreach(@rangestack)    { unless($_ !~ /\s+?/) { push(@thisItemsselected,$_); }}
1146    foreach(@singlestack)   { unless($_ !~ /\s+?/) { push(@thisItemsselected,$_); }}
1147
1148    ## SPECIAL POST EXTRACTION CASE RULES
1149
1150    my $FIND_BUY_ALL = 0;
1151
1152    foreach(@GBuyAllIndicators) { s/[\n\r]//g; next if($_ !~ /\s+?/); if($thisPreExtractionLine =~ /$_/) { $FIND_BUY_ALL = 1; } }
1153
1154    if ($FIND_BUY_ALL) {
1155
1156        if($exit) {
1157
1158            my $t = '.vfyTok';
1159            $eventType = 'WRN_FUP'; ($eventRequiresFUP !~ /$t/) ? $eventRequiresFUP .= $t : $eventRequiresFUP ; $FUPcount++; $excpCount++; $e
ventDesc = "  \<".$thisRecordID."\>  checkMultiBuy_Extract :: AMBIGUOUS ALL-ITEM SELECTION DETECTED IN EXTRACTION LOCATION <$pos>"; &logEvent(\*FH_log, $e
ventType, $eventDesc, $Cuser, $LGDEBUG_FTN);
1160
1161            push(@thisItemsselected," ALL ($thisMailingID) "); $exit=1;
1162
1163            }
1164
1165
1166        }
```

F:\Development\PRJ\PRJ_hbsp_extractorOrder--WORKING\procOrder.pl
Printed at 19:48 on 06 Feb 1999
Page 19

```
1167  ## DETECT DIFFERENCE IN EXTRACTION STACK AND MAX EXTRACTION STACK
1168
1169    if ($exit && (scalar(@singlestack) != scalar(@testsinglestack) ) ) {
1170
1171
1172      my $t = '.vfyTok'; $eventType = "WRN_FUP"; ($eventRequiresFUP !~ /$t/) ? $eventRequiresFUP .= $t: $eventRequiresFUP ; $FUPcount++; $excpCount++; $eventDesc
1173  = "\<".$thisRecordID."\>"  checkMultiBuy_Extract :: AMBIGUOUS EXTRACTION WARNING / INSPECT IN EXTRACTION LOCATION <$pos>"; &logEvent(\*FH_log, $eventType
1174  , $eventDesc, $Guser, $LGDEBUG_FTN);
1175
1176      if ($FIND_ADD_CHANGE) {
1177
1178
1179      foreach(@GaddresschangeIndicators) { s/[\n\r]//g; next if($_ !~ /\s+?/); if($thisPreExtractionLine =~ /$_/) { $FIND_ADD_CHANGE = 1; } }
1180
1181      }
1182      my $t = '.chkAC';
1183      $eventType = "WRN_FUP"; ($eventRequiresFUP !~ /$t/) ? $eventRequiresFUP .= $t : $eventRequiresFUP ; $FUPcount++; $excpCount++; $eventDesc
1184  = "\<".$thisRecordID."\>"  checkMultiBuy_Extract :: POSSIBLE ADD/CHANGE REQUEST DETECTED IN EXTRACTION LOCATION <$pos>"; &logEvent(\*FH_log, $eventType,
1185  $eventDesc, $Guser, $LGDEBUG_FTN);
1186
1187      }
1188      my $FIND_QUESTION = 0;
1189      if($thisPreExtractionLine =~ /\?\s/) { $FIND_QUESTION = 1; }
1190
1191
1192      if ($FIND_QUESTION) {
1193
1194      my $t = '.chkQst';
1195      $eventType = "WRN_FUP"; ($eventRequiresFUP !~ /$t/) ? $eventRequiresFUP .= $t : $eventRequiresFUP ; $FUPcount++; $excpCount++; $eventDesc
1196  = "\<".$thisRecordID."\>"  checkMultiBuy_Extract :: POSSIBLE ADD/CHANGE OR CARE REQUEST DETECTED IN EXTRACTION LOCATION <$pos>"; &logEvent(\*FH_log, $ev
1197  entType, $eventDesc, $Guser, $LGDEBUG_FTN);
1198
1199      }
1200      my $FIND_AI_ONLY = 0;
1201
1202      if ($exit && scalar(@singlestack) == 2 && scalar(@testsinglestack) == 2) {
1203
1204      ##  $x = scalar(@singlestack);
1205      ##  $y = scalar(@testsinglestack);
1206
1207      ##print "-- debug -- AI TEST __ singlestacklen = $x __ testsinglestacklen = $y \n";
1208
1209
1210      if(grep /(A)|(I)/, @singlestack) { $FIND_AI_ONLY = 1;}
1211
1212      }
1213      if ($FIND_AI_ONLY) {
1214
1215      my $t = '.vfyTok';
1216      $eventType = "WRN_FUP"; ($eventRequiresFUP !~ /$t/) ? $eventRequiresFUP .= $t : $eventRequiresFUP ; $FUPcount++; $excpCount++; $eventDesc
1217  = "\<".$thisRecordID."\>"  checkMultiBuy_Extract :: AMBIGUOUS EXTRACTION WARNING: \"A\" or \"I\" ONLY DETECTED"; &logEvent(\*FH_log, $eventType, $eventD
1218  esc, $Guser, $LGDEBUG_FTN);
1219
1220      }
1221      my $LIKELY_PARSE_ERRORS = 0;
1222      if ($exit && $thisPreExtractionLine =~ /[\[\]]+/) { $LIKELY_PARSE_ERRORS = 1; }
1223
1224
1225      my $t = '.vfyTok'; $eventType = "WRN_FUP"; ($eventRequiresFUP !~ /$t/) ? $eventRequiresFUP .= $t : $eventRequiresFUP ; $FUPcount++; $eventDesc
      = "\<".$thisRecordID."\>"  checkMultiBuy_Extract :: AMBIGUOUS EXTRACTION WARNING: BRACKETS FOUND IN EXTRACTION LOCATION <$pos>"; &logEvent(\*FH_log, $ev
      $eventDesc, $Guser,
```

```
1226
1227
1228
1229
1230       ##print "\n-$thisRecordID- debug singlestack -- \n";
1231       ##print @singlestack;
1232       ##print "\n-- debug testsinglestack -- \n";
1233       ##print @testsinglestack;
1234       ##print "\n-$thisRecordID------------------------------
1235       ##print "\n-$thisRecordID------------------------------\n";
1236
1237       }
1238       return $exit;
1239     }
1240     #
1241     #
1242     # -------------------------------------------------------
1243     #
1244     sub detectBuyTokens {
1245
1246       my($MULTI_BUY_ON, $Gsubj_buyToken,$thissubject,$thismailingID,$thisBodyTop,$thisBodyBottom,$ptrHASHLAYOUT,$ptrSCALARtopLines,$ptrSCALARbottomLines,$GDEBUG)
          = @_;
1247
1248       my %LAYOUT              = %$ptrHASHLAYOUT;
1249       my @topLines            = @$ptrSCALARtopLines;
1250       my @bottomLines         = @$ptrSCALARbottomLines;
1251       my $buyTokenPosition    = 'none';
1252       $thisBuyStatus          = undef;
1253       @thisItemsselected      = undef;
1254       $thisItemsSelectedstack = undef;
1255
1256       my $exit = 0;
1257
1258       ## print "--debug-- buyToken= |$Gsubj_buyToken|\n";
1259
1260       FIND_BUY_TOKENS: {
1261
1262       ##
1263       ## &checkMultiBuy_Extract returns success on a good match/extraction;
1264       ## @thisItemsselected is created/populated at this time as well
1265       ##
1266
1267       ## check $thissubject
1268
1269       ## print "\n\n=========debug-- SUBJECT = |$thissubject| \n";
1270       if(!$MULTI_BUY_ON && $thissubject =~ /$Gsubj_buyToken/) {
1271           $buyTokenPosition = ' S '; $thisBuyStatus = ('<'. $buyTokenPosition.'> |'. $thissubject. '|');
1272
1273       ## print "--debug-- thisBuyStatus= |$thisBuyStatus| \n";
1274
1275           $buyTokenPosition = ' S '; $thisBuyStatus = ('<'. $buyTokenPosition.'> |'. $thissubject. '|');
1276
1277           last FIND_BUY_TOKENS;
1278       }
1279
1280       if($MULTI_BUY_ON) {
1281           if(&checkMultiBuy_Extract($thissubject,$thismailingID,' S ',\%LAYOUT, $GDEBUG)) {
1282               $buyTokenPosition = ' S '; $thisBuyStatus = ' <'. $buyTokenPosition.'>';
1283
1284               last FIND_BUY_TOKENS;
1285           }
1286       }
1287
1288
1289
1290
1291       ## check @topLines
1292       ## print "\n\n=======debug-- thisBodyTop = |$thisBodyTop| \n";
1293
```

```
1294   if(!$MULTI_BUY_ON) {
1295
1296     foreach(@topLines) {
1297
1298       $thisTopLine = $_;
1299
1300       if($thisTopLine =~ /$Gsubj_buyToken/) {
1301
1302         $buyTokenPosition = ' T ';
1303         $thisBuyStatus = '<'.$buyTokenPosition.'> '.$thisTopLine;
1304
1305         ## print "--debug-- thisBuyStatus= |$thisBuyStatus| \n";
1306
1307
1308
1309       }
1310
1311       last FIND_BUY_TOKENS;
1312     }
1313
1314   if($MULTI_BUY_ON) {
1315
1316     if(&checkMultiBuy_Extract($thisBodyTop,$thismailingID,' T ',\%LAYOUT,$GDEBUG)) {
1317
1318       $exit =1;
1319
1320         $buyTokenPosition = ' T ';
1321         $thisBuyStatus = '<'.$buyTokenPosition.'> ';
1322
1323
1324
1325     }
1326
1327       last FIND_BUY_TOKENS;
1328
1329   }
1330   ## check @bottomLines
1331
1332   ## print "\n\n======debug-- thisBodyBottom = |$thisBodyBottom| \n";
1333
1334   if(!$MULTI_BUY_ON) {
1335
1336     foreach(@bottomLines) {
1337
1338       $thisBottomLine = $_;
1339
1340       if($thisBottomLine =~ /$Gsubj_buyToken/) {
1341
1342         $buyTokenPosition = ' B ';
1343         $thisBuyStatus = '<'.$buyTokenPosition.'> '.$thisBottomLine;
1344
1345         ## print "--debug-- thisBuyStatus= |$thisBuyStatus| \n";
1346
1347
1348       }
1349
1350       last FIND_BUY_TOKENS;
1351     }
1352
1353   if($MULTI_BUY_ON) {
1354
1355     if(&checkMultiBuy_Extract($thisBodyBottom,$thismailingID,' B ',\%LAYOUT,$GDEBUG)) {
1356
1357       $exit =1;
1358
1359         $buyTokenPosition = ' B ';
1360         $thisBuyStatus = '<'.$buyTokenPosition.'> ';
1361
1362     }
```

```perl
1363
1364
1365
1366
1367            last FIND_BUY_TOKENS;
1368          }
1369        }
1370
1371  ## print "--debug-- ===== $i ====== buyTokenPosition= |$buyTokenPosition|\n";
1372
1373      return $buyTokenPosition;
1374    }
1375
1376  #
1377  #
1378  #
1379  # ----------------------------------------
1380  sub extractDedupItemsselectedstack {
1381    my(@thisItemsselected) = @_;
1382
1383    my $thisItemsselectedstack = undef;
1384    my $thisItemsselectedstackseparator = ',';
1385    my $i = undef;
1386
1387    for($i = 0; $i <= $#thisItemsselected; $i++) {
1388
1389      my $item = $thisItemsselected[$i];  # all items must be normalized!
1390
1391      next if($item !~ /\S+?/);
1392
1393      $item=&normalizespacing($item);
1394
1395      $thisItemsselectedstack .= ($item.$thisItemsselectedstackseparator);
1396
1397    }
1398
1399    $thisItemsselectedstack =~ s/\,\s*$//;
1400
1401    return  $thisItemsselectedstack;
1402  }
1403
1404  #
1405  #
1406  #
1407  # ----------------------------------------
1408  sub getDatestamp {
1409
1410  # get standardized date stamp (UNIX, WINNT)
1411  # returns a date stamp
1412
1413  my($Gplatform,$LGDEBUG_FTN) = @_;
1414
1415  if(!defined($Gplatform)) { $Gplatform = 'WIN'; }
1416    my $GTIME    = '';
1417    my $day      = undef;
1418    my $val      = undef;
1419
1420    if ($Gplatform ne 'WIN') { $val = `date /t`; }
1421      else {
1422        $val = `date /t`;
1423        chop($val);
1424        $val =~ /(\w+)\s+(\d+)\/(\d{2})(\d{2})/;
1425        $val = $5.$2.$3; $day = $1;
1426      }
1427
1428    if ($Gplatform eq 'WIN') {
1429      $GTIME = `echo '' | time`;
1430      $GTIME =~ /\:\s*(\d{1,2})\:(\d{2})\:(\d{2})\.(\d{2})/;
1431
```

```perl
1432
1433
1434
1435
1436
1437         my $leadingDigit = $1;
            if (length($leadingDigit) == 1) { $leadingDigit = ('0'.$leadingDigit); }
            $val .= $leadingDigit.$2.$3.$4.'.'.$day;
1438
        }
1439 if($LGDEBUG_FTN) { print "DEBUG_gdate=".$val."\n"; }
1440 return $val;
1441 }
1442 #
1443 #
1444 #
1445 #
1446 #
1447 sub getfieldkeys {
1448
1449 my($ptrLayout) = @_;
1450 my(%LAYOUT) = %$ptrLayout;
1451
1452 ## prepare and count required fields from body data layout for check against body
1453
1454 my(@fieldkeys)   = undef;
1455 my($k)           = undef;
1456
1457 ##$f = scalar(@fieldkeys); print "--debug-- scalarstartFtn=$f \n";
1458
1459 foreach $k(sort keys %LAYOUT) {
1460
1461     next if($k !~ /\[a-z]{1,2}_/);   ## these elements are not fieldnames but are other CF parameters (name \t value)
1462
1463     ## print "--debug-- k= |$k|\n";
1464
1465     if($k =~ /\S+?/) { push(@fieldkeys,$k); }
1466 }
1467
1468 shift @fieldkeys;
1469 ## $f = scalar(@fieldkeys); print "--debug-- scalarEndFtn=$f \n";
1470
1471 return @fieldkeys;
1472 }
1473
1474 #  ----------------------------------------------------------------
1475
1476 sub getkeyPos_outfile {
1477
1478 my($ptrHASHrecordTemplate) = @_;
1479
1480 my(%HASHrecordTemplate) = %$ptrHASHrecordTemplate;
1481
1482 my @record                                         = undef;
1483 $SalutationFieldpos          = -1;
1484 $IDFieldpos                  = -1;
1485 $LIDFieldpos                 = -1;
1486 $MailingIDFieldpos           = -1;
1487 $firstNameFieldpos           = -1;
1488 $lastNameFieldpos            = -1;
1489 $ixFieldpos                  = -1;
1490 $titleFieldpos               = -1;
1491 $stateFieldpos               = -1;
1492 $countryFieldpos             = -1;
1493 $postalCodeFieldpos          = -1;
1494 $emailFieldpos               = -1;
1495 $companyFieldpos             = -1;
1496 $addressFieldpos             = -1;
1497 $departmentFieldpos          = -1;
1498 $cityFieldpos                = -1;
1499 $fromAddressFieldpos         = -1;
1500
```

F:\Development\PRJ\PRJ_hbsp_extractOrder--WORKING\procOrder.pl
Page 24
Printed at 19:48 on 06 Feb 1999

```perl
1501   my $i                = 0;
1502
1503   @record = sort(keys(%HASHrecordTemplate));
1504
1505   ##print "---DEBUG---"; print @record; sleep 4;
1506
1507   for($i=0; $i <= $#record; $i++) {
1508
1509       $record[$i] = uc($record[$i]);
1510
1511
1512       CASE: {
1513           if ($record[$i] =~ /FROM\_ADDRESS/)                                        { $fromAddressFieldPos = $i; last CASE; }
1514           if ($record[$i] =~ /(EMAIL)|(E-MAIL)|(INTERNET_ADDR)|(INTERNET_AD)/)       { $emailFieldPos       = $i; last CASE; }
1515           if ($record[$i] =~ /(COUNTRY)|(CTRY)/)                                     { $countryFieldPos     = $i; last CASE; }
1516           if ($record[$i] =~ /(ZIP)|(POSTAL)/)                                       { $postalCodeFieldPos  = $i; last CASE; }
1517           if ($record[$i] =~ /(INST)|(COMPANY)|(ORGANIZATION)/)                      { $companyFieldPos     = $i; last CASE; }
1518           if ($record[$i] =~ /DEPARTMENT/)                                           { $departmentFieldPos  = $i; last CASE; }
1519           if ($record[$i] =~ /(ADDR)|(ADDRESS)/)                                     { $addressFieldPos     = $i; last CASE; }
1520           if ($record[$i] =~ /CITY/)                                                 { $cityFieldPos        = $i; last CASE; }
1521           if ($record[$i] =~ /CUSTNO/)                                               { $LIDFieldPos         = $i; last CASE; }
1522           if ($record[$i] =~ /LISTID/)                                               { $LISTIDFieldPos      = $i; last CASE; }
1523           if ($record[$i] =~ /(PRIORITY)/)                                           { $mailingIDFieldPos   = $i; last CASE; }
1524           if ($record[$i] =~ /(.*F)|(MAILING)/)                                      { $firstNameFieldPos   = $i; last CASE; }
1525           if ($record[$i] =~ /(.*L\w+NAME.*)|(FIRST)/)                               { $lastNameFieldPos    = $i; last CASE; }
1526           if ($record[$i] =~ /A.*IX.*$/)                                             { $IxFieldPos          = $i; last CASE; }
1527           if ($record[$i] =~ /TITLE/)                                                { $titleFieldPos       = $i; last CASE; }
1528           if ($record[$i] =~ /STATE/ || $record[$i] =~ /\_CS/)                       { $stateFieldPos       = $i; last CASE; }
1529       };
1530   }
1531
1532       if ($firstnameFieldPos == -1)   { print STDOUT "[HEADER] WARNING: \/\w+F\|\w+NAME\/ was not found!\n"; }
1533       if ($lastnameFieldPos  == -1)   { print STDOUT "[HEADER] WARNING: \/\w+L\|\w+NAME\/ was not found!\n"; }
1534       if ($firstnameFieldPos == -1)   { print STDOUT "[HEADER] WARNING: \/\w+*IX\/ was not found!\n"; }
1535       if ($titleFieldPos     == -1)   { print STDOUT "[HEADER] WARNING: \/TITLE\/ was not found!\n"; }
1536       if ($stateFieldPos     == -1)   { print STDOUT "[HEADER] WARNING: \/STATE\/ was not found!\n"; }
1537       if ($postalcodeFieldPos == -1)  { print STDOUT "[HEADER] WARNING: \/ZIP|POSTAL\/ was not found!\n"; }
1538       if ($countryFieldPos   == -1)   { print STDOUT "[HEADER] WARNING: \/(COUNTRY)|(CTRY)\/ was not found!\n"; }
1539       if ($emailFieldPos     == -1)   { print STDOUT "[HEADER] WARNING: \/(EMAIL)|(E-MAIL)|(INTERNET_ADDR)|(INTERNETAD)\/ was not found!\n";
1540   }
1541
1542
1543   #
1544   # ---------------------------------------------------------------------
1545   #
1546
1547
1548   sub getnextDBkey_return {
1549
1550       my($keyFilename, $LGDEBUG_FTN) = @_;
1551
1552       $LOCK_SH = 1;
1553       $LOCK_EX = 2;
1554       $LOCK_NB = 4;
1555       $LOCK_UN = 8;
1556
1557       $POS_BOF = 0;
1558       $POS_CUR = 1;
1559       $POS_EOF = 2;
1560
1561       $/ = undef;
1562       $startkeys = 0;
1563       $startkeys = 0;
1564       if(! -s $keyFilename) { `echo $startkeys >> $keyFilename`; }
1565
1566       OPEN_DB: if (!open(KEYS, "+< $keyFilename")) {$exit = 0; $eventType = 'FAIL'; $eventDesc = "getNextDBkey_return::Cannot open: \[$keyFilename \]"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);}
1567
```

F:\Development\PRJ\PRJ_hbsp_extractorOrder--WORKING\procOrder.pl
Printed at 19:48 on 06 Feb 1999
Page 25

```perl
1568      flock(KEYS, $LOCK_EX);
1569
1570          my($count)          = <KEYS>;
1571          $nextkey           = $count+1;
1572          seek(KEYS,0,$POS_BOF);
1573          print KEYS $nextkey;
1574
1575      flock(KEYS, $LOCK_UN);
1576      close(KEYS);
1577
1578      return $nextkey;
1579  }
1580  #
1581  # -----------------------------------------------------------------
1582  #
1583  #
1584  #
1585  sub HASHloadFieldmap {
1586
1587      my($filename, $LGDEBUG_FTN) = @_;
1588
1589      if (!open(FH, "$filename")) { $eventType = 'FAIL'; $eventDesc = "HASHloadFieldmap::Cannot open: \[$keyFilename \]"; &logEvent(\*FH_log, $event
          Desc, $Guser, $LGDEBUG_FTN);}
1590
1591      $/ = "\n";
1592
1593      my(@fieldmapTemp) = <FH>;
1594      my($key)   = undef;
1595      my($value) = undef;
1596      my(%fieldmap) = undef;
1597
1598      close(FH);
1599
1600      foreach (@fieldmapTemp) {
1601
1602          my $item = $_;
1603
1604          next if($item =~ /\/\/) || ($item !~ /\s+?/); # comments
1605
1606          ($key, $value) = split(/[\t\,]/, $item);
1607
1608          $value =~ s/[\n\r]//g;
1609
1610          $fieldmap{$key} = $value;
1611
1612      }
1613
1614      return %fieldmap;
1615  }
1616  #
1617  # -----------------------------------------------------------------
1618  #
1619  #
1620
1621  sub loadFile_SCALAR {
1622
1623      my($filename) = @_;
1624
1625      if (!open(FILE, "$filename")) {$eventType = 'FAIL'; $eventDesc = "loadArray::Cannot open filename: \[$filename\]"; &logEvent(\*FH_log, $eventType, $eventD
          esc, $Guser, $LGDEBUG_FTN);}
1626
1627      $/ = "\n";
1628
1629      my(@all) = undef;
1630
1631      while(<FILE>) {
1632
1633          my $buf = $_;
1634          next if($buf !~ /\s+?/ || $buf =~ /\/\/);
```

```perl
1635            push(@all, $buf);
1636        }
1637
1638
1639    return @all;
1640  }
1641
1642  #
1643  #
1644  #
1645  # -----------------------------------------
1646  sub logEvent {
1647
1648      my($FH,$eventType,$eventDesc,$Guser,$LGDEBUG_FTN) = @_;
1649
1650      my $dateNow = &getDatestamp('WIN',$LGDEBUG_FTN);
1651      # print event to STDOUT
1652
1653      if($eventType eq 'ABORT') {
1654
1655          $foo = `mapisend -u administrator -p password -r aestes\@e-dialog.com -s "** $dateNow procorder - critical error"   -m "$Guser $GClientName $eventType $eventDesc";
1656          die "Error! Cannot log! "$eventType."\t".$eventDesc."\n" ;
1657      }
1658
1659      if($eventType !~ /_x_/) { print $eventType."\t".$eventDesc."\n"; }
1660
1661          else { $eventType =~ s/_x_//; }
1662
1663  # write event to activity log
1664
1665      &writeRecord(\*FH_log,$GfieldDelimiter,$thisLogkey=&getNextDBkey_return($GactivityLogkeysFilename,$LGDEBUG_FTN),\%GactivityLogRecord,'',$LGDEBUG_F
1666  TN);
1667
1668      if($eventType eq 'FAIL') {
1669
1670          $foo = `mapisend -u administrator -p password -r aestes\@e-dialog.com -s "** $dateNow procorder - critical error"   -m "$Guser $GClientName
1671          $eventType $eventDesc";
1672              exit;
1673      }
1674  #
1675  #
1676  #
1677  # -------------------------------------------
1678  sub normalizeSpacing {
1679
1680  my($dirty) = @_;
1681
1682  $dirty =~ s/\s{2,}/$space/g; $dirty =~ s/\s+$//; $dirty =~ s/^\s+//;
1683  $clean = $dirty;
1684
1685  return $clean
1686  }
1687  #
1688  #
1689  #
1690  # -------------------------------------------
1691
1692  sub queryUPDATE_DB {
1693
1694  my($ptrLAYOUT,$thisQuerykey, $thisQueryFieldName, $fieldName, $fieldvalue, $fieldName1, $fieldvalue1, $RECORD_RECOVERED, $LGDEBUG_FTN) = @_;
1695
1696  ##print "** DSN = $LAYOUT{'thismailingDSN'} \n";
1697
1698  my(%LAYOUT) = %$ptrLAYOUT;
1699  my $RS = undef;
1700  my $Conn = undef;
```

```perl
1701   my $Errors = undef;
1702   my @fieldnames = undef;
1703   my $PID = -1;
1704   my $recCount = 0;
1705   my $EXIT = 1;
1706
1707   # construct/open DSN
1708
1709   my $Conn = undef;
1710
1711   $Conn = new Win32::OLE("ADODB.Connection");
1712   $Conn->Open($LAYOUT{'thisMailingDSN'},"","");
1713
1714   my(%DB) = undef;
1715
1716   # Load data into recordset
1717
1718        my $mailingTable = $LAYOUT{'thisMailingTable'};
1719        my %foo = undef;
1720
1721   # look to see if update record exists, $thisQueryKey
1722
1723        unless($RECORD_RECOVERED) {
1724
1725             my $RECORD_EXISTS = 1;
1726             %foo = &querySELECT_DB(\%LAYOUT,$thisQueryFieldname,$thisQueryKey,$LGDEBUG_FTN);
1727             if($foo{'_exit_'} < 1) { $RECORD_EXISTS = 0; $EXIT = 0; return $EXIT; }
1728
1729        }
1730
1731        $thisQuery = "UPDATE $mailingTable SET $fieldname = \'$fieldvalue\', $fieldname1 = \'$fieldvalue1'   WHERE $thisQueryFieldname = \'$thisQueryKey\'
1732
1733        $RS = $Conn->Execute($thisQuery);
1734   ##%foo = %$RS;
1735   ##foreach(keys(%foo)) { print "--key = $_ __ value = $foo{$_} \n"; }
1736
1737        if(!$RS) {
1738
1739             $Errors = $Conn->Errors();
1740             print "Errors:\n";
1741             foreach $error (keys %$Errors) {
1742                  print $error->{Description}, "\n";
1743             }
1744             $eventType = 'WRN_FUP'; $eventRequiresFUP .= (($eventRequiresFUP =~ /badMDB/)?'':';'.badMDB'); $eventDesc = " \<".$thisRecordID."\>   ($LAYO
1745   UT{'thisMailingDSN'}) queryUPDATE_DB :: Cannot create RS: \[$thisQuery\]"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
1746             $EXIT = 0;
1747             return $EXIT;
1748        }
1749
1750        $eventType = 'INFO'; $eventDesc = " \<".$thisRecordID."\>         ($LAYOUT{'thisMailingDSN'}) queryUPDATE_DB :: UPDATE SUCCESSFUL: \[$thisQuery\]"; &log
1751   Event(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_FTN);
1752   return $EXIT;
1753   }
1754
1755   #
1756   #
1757   # -------------------------------------------------
1758   #
1759   sub querySELECT_DB {
1760
1761   my($ptrLAYOUT,$thisQueryFieldname,$thisQueryKey,$LGDEBUG_FTN) = @_;
1762
1763   ##print "** DSN = $LAYOUT{'thisMailingDSN'} \n";
1764
1765   my(%LAYOUT) = %$ptrLAYOUT;
1766   my $RS = undef;
```

F:\Development\PRJ\PRJ_hbsp_extractorOrder--WORKING\procOrder.pl
Printed at 19:48 on 06 Feb 1999
Page 28

```perl
1767 my  $conn = undef;
1768 my  $errors = undef;
1769 my  @fieldnames = undef;
1770 my  $PID = -1;
1771 my  $recCount = 0;
1772
1773 # construct/open DSN
1774
1775 my  $conn = undef;
1776
1777 $conn = new win32::OLE("ADODB.Connection");
1778 $conn->open($LAYOUT{'thismailingDSN'},"","");
1779
1780 my(%DB) = undef;
1781 $DB{'_exit_'} = 1;
1782
1783 # Load data into recordset
1784
1785 my $mailingTable = $LAYOUT{'thismailingTable'};
1786
1787 $thisquery = "SELECT * FROM $mailingTable WHERE $thisqueryFieldname = \'$thisqueryKey\'";
1788
1789 $RS = $conn->Execute($thisquery);
1790
1791 if(!$RS) {
1792
1793       $errors = $conn->Errors();
1794       print "errors:\n";
1795       foreach $error (keys %$errors) {
1796       print $error->{Description}, "\n";
1797
1798       $eventType = 'FAIL'; $eventDesc = " \<".$thisRecordID."\>    ($LAYOUT{'thismailingDSN'}) querySELECT_DB :: Cannot create RS: \[$thisquery\]"; &loge
1799 vent(\*FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);
1800
1801 my  $i = undef;
1802
1803 for($i=0; $i < $RS->Fields->Count; $i++) { $fieldnames[$i] = $RS->Fields->Item($i)->Name; }
1804
1805 my $RESULT_FLAG = 1;
1806
1807 if($RS->EOF) { $RESULT_FLAG = 0; }
1808 $RS -> MoveNext;
1809
1810 if(!$RS->EOF) { $RESULT_FLAG = -1; }
1811 $RS -> MovePrevious;
1812
1813 if($RESULT_FLAG < 1) {
1814
1815 (($eventRequiresFUP =~ /badMDB/)?'':';badMDB'); $FUPcount++; $eventType = 'ERR_FUP'; $eventDesc = " \<".$thisRecordID."\>    ($LAYOUT{'thismailingD
1816 SN'}) querySELECT_DB :: Bad Lookup ($RESULT_FLAG): \[$thisquery\]"; &logEvent(\*FH_log, $eventType, $eventDesc, $guser, $LGDEBUG_FTN);
1817
1818 $DB{'_exit_'} = $RESULT_FLAG;
1819
1820 return %DB;
1821
1822 while ( !$RS->EOF ) {
1823
1824       for($i=0; $i < $RS ->Fields->Count; $i++) {
1825
1826             $DB{$fieldnames[$i]}= uc($RS->Fields->Item($i)->value);
1827
1828       }
1829
1830       $RS->MoveNext; $recCount++;
1831
1832 }
1833 $conn->Close;
```

```perl
1834     return %DB;
1835 }
1836
1837 #
1838 # --------------------------------------------------------------------------------
1839 #
1840
1841 sub read_DSNtoAOH {
1842
1843 my($DSN, $LGDEBUG_FTN) = @_;
1844
1845 ##print "** DSN = $DSN \n";
1846
1847 $RS = undef;
1848 $Conn = undef;
1849 $Errors = undef;
1850 @fieldNames = undef;
1851 $PID = -1;
1852 $recCount = 0;
1853
1854 # construct/open DSN
1855
1856 $Conn = new Win32::OLE("ADODB.Connection");
1857 $Conn->Open($DSN,"","");
1858 my(@FILE) = undef;
1859
1860 # load data into recordset
1861
1862 $thisQuery = "SELECT * FROM Email";
1863
1864 $RS = $Conn->Execute($thisQuery);
1865
1866 if(!$RS) {
1867     $Errors = $Conn->Errors();
1868     print "Errors:\n";
1869     foreach $error (keys %$Errors) {
1870         print $error->{Description}, "\n";
1871     }
1872     $eventType = 'FAIL'; $eventDesc = "read_DSNtoAOH::Cannot create RS: \[$thisQuery\]"; &logEvent(\*FH_log, $eventType, $eventDesc, $Guser, $LGDEBUG_
1873 FTN);
1874
1875 my $i = undef;
1876
1877 for($i=0; $i < $RS->Fields->Count; $i++) { $fieldNames[$i] = $RS->Fields->Item($i)->Name; }
1878
1879 while ( !$RS->EOF ) {
1880
1881     for($i=0; $i < $RS ->Fields->Count; $i++) {
1882         $FILE[$recCount]{$fieldNames[$i]}= $RS->Fields->Item($i)->value;
1883     }
1884
1885 }
1886
1887     $RS->MoveNext; $recCount++; if ($recCount % 20 == 0) { print STDOUT "+"; }
1888 }
1889
1890 $Conn->Close;
1891
1892 return @FILE;
1893 }
1894
1895 #
1896 # --------------------------------------------------------------------------------
1897 #
1898
1899 sub removeTabs {
1900
1901 my($tabs) = @_;
```

```
1902            $tabs = s/\/t//g;
1903
1904    return $tabless
1905
1906    }
1907
1908    #
1909    #
1910    #
1911    # ----------------------------------------------------------------
1912
1913    sub subjectSynonymnLookup {
1914
1915        my($subject,$ptrHASHLAYOUT) = @_;
1916        my(%LAYOUT) = %$ptrHASHLAYOUT;
1917        my($thismailingID) = -1;
1918
1919        $subject =~ /\s*([|=\*|\+\|-]{1,3})\s*/;
1920        $thissubjectmailingIDToken = $1;
1921
1922        if(defined($LAYOUT{$thissubjectmailingIDToken})) { $thismailingID = $LAYOUT{$thissubjectmailingIDToken}; }
1923
1924        ## print "-- debug -- ATTEMPT MAILING ID RECOVERY... TOKEN= |$thissubjectmailingIDToken|... RESULT MAILINGID= |$thismailingID|\n";
1925
1926    return $thismailingID
1927    }
1928
1929    #
1930    #
1931    #
1932    # ----------------------------------------------------------------
1933
1934    sub writeRecord {
1935
1936    # takes FH by ref; has _e_ feature for values
1937    my($FH,$GfieldDelimiter,$logkey,$ptrHASHrecord,$RECORD_TYPE, $LGDEBUG_FTN) = @_;
1938
1939    flock($FH, $LOCK_EX);
1940
1941    my $fileEmpty = 0;
1942    if(-z $FH) {$fileEmpty = 1;}
1943
1944    %HASHrecord = %$ptrHASHrecord;
1945
1946    $space = " ";
1947    $exit = 1;
1948
1949    $LOCK_SH = 1;
1950    $LOCK_EX = 2;
1951    $LOCK_NB = 4;
1952    $LOCK_UN = 8;
1953
1954    my $record = undef;
1955    my $header = undef;
1956    my $key = undef;
1957
1958    foreach $key(sort(keys(%HASHrecord))) {
1959
1960        $value = $HASHrecord{$key};
1961
1962        if($fileEmpty) { $key =~ s/[a-z]{1,2}-//; $header .= (uc($key).$GfieldDelimiter); }
1963
1964        if($value =~ /^_e_(.{2,})$/) { $value = eval($1); }
1965
1966
1967        $record .= ($value.$GfieldDelimiter);
1968
1969    }
1970    chop($record);                                          # remove last field delimiter
```

```
1971
1972 if($fileEmpty) { chop($header); print $FH $header."\n"; }
1973 if($ADD_CHANGE_ON && ${$RECORD_TYPE ne 'BODY') { $record = &canonicalize($record); }
1974 print $FH $record."\n";
1975
1976 flock($FH, $LOCK_UN);
1977 }
1978
1979 __END__
1980
```